**Readme**

**Part 1.**

The solution of this task is divided into two sections:

1. **Explorative data analysis and model training.**

All the details including the code, data analysis, plots and model evaluations are accessible in the **EDA&ModelTraining.ipynb** file.

1. **Product-like code to generate the daily prediction and the relevant unit tests**

The source files: **prediction.py** and **prediction\_test.py**

After analyzing the data, we can summarize:

* The data is relevant clean and there’s no missing value
* Some features (registered and casual count) are in the reality would not be given when predicting. The sum of them is the to be predicted count (‘cnt’). So that could explain why these two also the highly correlate to the ‘cnt’.
* It contains both numerical and categorical features and fortunately the categorical features are already labeled
* The to be predicted target shows some periodicity along time. That gives a clue that the trained regression model might have a good result evaluation.

Before training the models, we need some preprocessing and feature engineering:

1. Drop the useless features
2. Chose which features as categorial (or as numerical)
3. Scale the numerical features
4. On hot encoding the categorical features

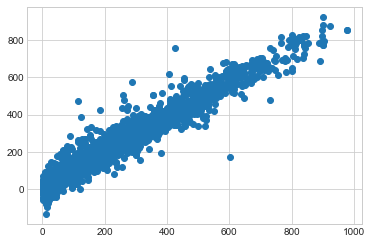
Definitely, this is a regression issue but there are lot of different models to address this problem. Considering the time and computing limitation, there are only a few to be experimented:

* The simplest linear regression model
* SVR
* Xgboost
* Lasso regression model

In order to find the best estimator, usually the grid search and cross validation should be implemented during training. Here only SVR is implemented by grid search with cross validation. Even with only 18 combinations of hyperparameters it is still time consuming on PC. For this reason, the Xgboost (using the similar addressed regression problem as reference for Xgboost) and Lasso is trained using kind of random picked hyperparameters.

Surprisingly, the Xgboost gives a quiet good result shown below (x- axis: true value; y – axis: predict value). The Xgboost is then selected as the best estimator. (All the trained estimator are persisted as ‘.joblib’ file.

|  |  |  |
| --- | --- | --- |
| Evaluations of Trained Xgboost | | |
| R square score | mean absolute err | explained variance score |
| 0.9485 | 28.1000 | 0.9485 |



Beside the selection of models and hyperparameter, the feature engineering is also a important issue or philosophy. Because of the resource limitation, here more experiments on feature engineering would not be executed. However we can still raise some options of this issue:

* Take more features as categorical and then do one-hot-encoding on them (e.g. month)
* Dimension reducing (e.g. PCA)

In order to produce a daily prediction, the class *Predictor* in *prediction* module (.py) implement the predicting using *pandas.DataFrame* to inisulize. The unit test to it is implemented by class *PredictorTest* in *prediction\_test* module (.py).

For simplification here assume that the input data has the same data type of each column and the same quality as the hour.csv (which means clean, no odd value, no missing value….), otherwise an extra unit should be implemented to check the quality of the input data.